THIS PAGE BLANK (USPTO)

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 3 : G06F 7/26; G11C 17/00

A1

(11) International Publication Number: WO 83/ 03912

(43) International Publication Date: 10 November 1983 (10.11.83)

(21) International Application Number: PCT/US82/00564

(22) International Filing Date: 30 April 1982 (30.04.82)

(71) Applicant (for all designated States except US): MASSACHUSETTS INSTITUTE OF TECHNOLOGY [US/US]; 77 Massachusetts Avenue, Cambridge, MA 02139 (US).

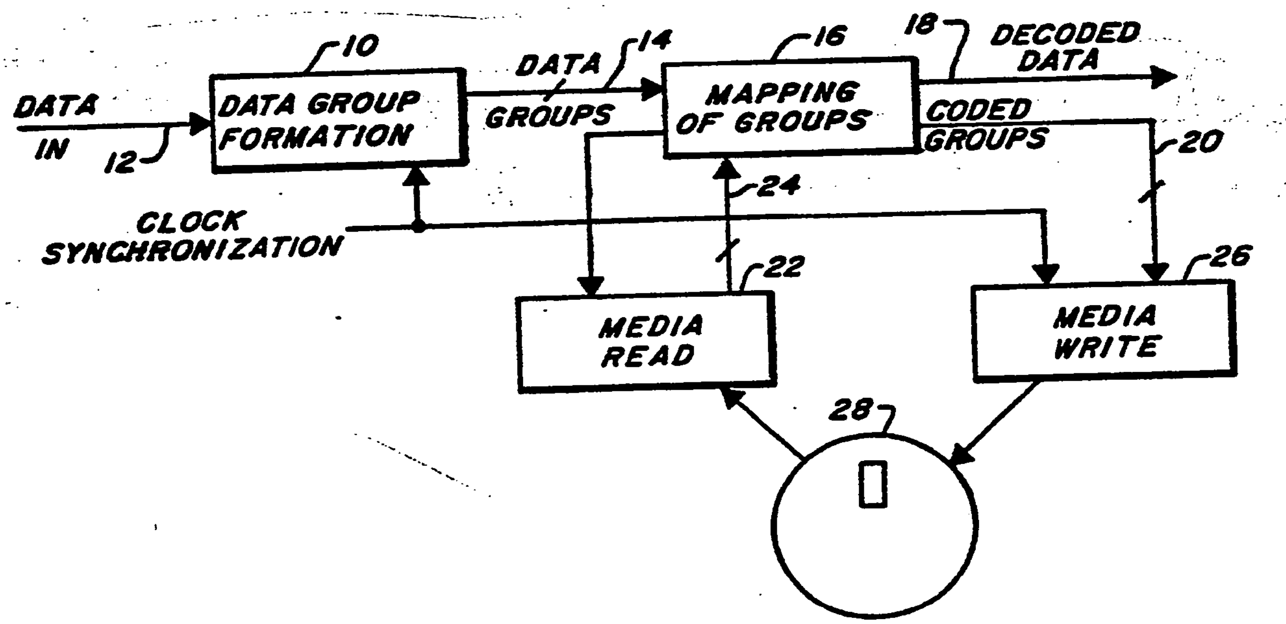
(72) Inventors; and
(75) Inventors/Applicants (for US only) : RIVEST, Ronald, L. [US/US]; 24 Candia Street, Arlington, MA 02174 (US). SHAMIR, Adi [IL/IL]; Applied Math Department, Weizmann Institute of Science, P.O.B. 26, Rehovot 76100 (IL).

(74) Agents: SMITH, Arthur, A., Jr.; Massachusetts Institute of Technology, Room E19-722, 77 Massachusetts Avenue, Cambridge, MA 02139 (US) et al.

(81) Designated States: AT (European patent), BE (European patent), CF (OAPI patent), CG (OAPI patent), CH (European patent), CM (OAPI patent), DE, DE (European patent), FR (European patent), GA (OAPI patent), GB, GB (European patent), JP, LU (European patent), NL (European patent), SE (European patent), SN (OAPI patent), TD (OAPI patent), TG (OAPI patent), US.

Published
With international search report.

(54) Title: METHOD AND APPARATUS FOR REUSING NON-ERASABLE MEMORY MEDIA



(57) Abstract

Storage media (28) such as digital optical disks, PROMS, and paper tape consist of a number of bit positions which initially contain a 'zero' and which can later be irreversibly overwritten with a 'one'. An apparatus and method provide for rewriting in such 'non-erasable memories (28)' for expanding their capacity by allowing the non-erasable memory (28) to be rewritten one or more times.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BE	Belgium
BR	Brazil
CF	Central African Republic
CG	Congo
CH	Switzerland
CM	Cameroon
DE	Germany, Federal Republic of
DK	Denmark
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
HU	Hungary
JP	Japan
KP	Democratic People's Republic of Korea
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
MC	Monaco
MG	Madagascar
MR	Mauritania
MW	Malawi
NL	Netherlands
NO	Norway
RO	Romania
SE	Sweden
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
US	United States of America

METHOD AND APPARATUS FOR REUSING NON-ERASABLE MEMORY MEDIA

The invention relates generally to a method and apparatus for writing in data storage systems and in particular to a method and apparatus for writing in a non-erasable storage medium such as a digital optical disk or paper tape.

Background of the Invention

Non-erasable, or write-once, data storage media such as digital optical disks, paper tape, PROMS, etc., have a plurality of bit positions, each of which can be irreversibly changed from an original state to a new state but one time. Thus, typically, the initial state of the bit position is designated a "zero" and the "zero" can be overwritten with a "one" when data is written into the media. Once the data has been written in a section of the media, which may be all of the media, that section is considered to be "used" by those experienced in the field, and is not considered to be reusable thereafter for recording new data.

Some non-erasable media, notably digital optical disks, can store vast amounts of data. A single twelve-inch disk can be used to store over 10^{11} bits of data, the equivalent of forty reels of magnetic tape, and access can be provided to any of it in about one-tenth of a second. This is an order of magnitude improvement in the cost/performance of memory technology and has had dramatic effects. And, while the cost of a high capacity, single twelve-inch disk may be only $100, that high capacity and low cost are achieved only at the further philosophical cost of making the writing process irreversible.

Similar limits exist, and are probably more familiar, in connection with the non-erasable media, such as punched paper tape, punched cards, and programmable read only memories (PROMS). However, the tremendous capacity and low cost per bit of digital optical disks provides a strong motivation for more closely examining their one drawback, the non-erasable nature of the memory storage process.

Therefore a primary object of this invention is an apparatus and method for reusing, that is writing more than once in, non-erasable memories. Other objects of the invention are an apparatus and method for reliably and simply increasing the effective capacity of non-erasable memories, and for providing a simple, easily implemented apparatus and method for rewriting in an otherwise non-erasable memory.

Summary of the Invention

In one aspect, the invention relates to apparatus for reusing a non-erasable memory medium. The apparatus features reading circuitry for reading successive groups of digits from the non-erasable memory medium and a process element for writing over each of the successive groups in accordance with both the input data information read at each successive group and respective new incoming input-data information. Thus, the process element provides an output in accordance with a predetermined "mapping" procedure or plan which is determined by the two data information sources recited above. Thereafter, the "written over" groups of digits can be uniquely read, at a later time, to reproduce the just written new incoming input-data information.

In another aspect, the invention relates to apparatus for writing a plurality of times in a non-erasable memory. The invention features receiving circuitry for receiving incoming digital data and for forming sequential input-data groups from the received data. A reading element reads successive groups of digital data from successive groups of data positions in the non-erasable memory. In response to each formed sequential input-data group and a respective read successive data group from memory, mapping circuitry generates successive new coded groups of digital data. Writing circuitry sequentially writes each of the coded groups of digital data over the corresponding read data group. Thereby, the coded groups of data can be later read and uniquely decoded to generate the previously received incoming digital data. Depending upon the coding or mapping method, the non-erasable medium can be rewritten a substantial number of times.

In yet another aspect, the invention relates to a method for writing a plurality of times in a non-erasable memory. The method features the steps of receiving input digital data, grouping the received input data into sequential groups of data, reading from the memory successive groups of stored digital data, forming new successive groups of write digital data from the stored digital data and the input data sequential groups, and writing the new groups of data over corresponding ones of the read groups from memory in a one-to-one relationship. The new groups of data are uniquely decodable to generate the input digital data.

In yet another aspect, the invention relates to a method for reusing a non-erasable memory medium.

This method features the steps of reading successive groups of digits from the non-erasable memory medium, and thereafter writing over each successive group in accordance with (a) the data values of new incoming data information and (b) the just read data information written at the successive positions to be overwritten. The new data written into the medium is generated according to a predetermined mapping method of said data inputs. Thereby, the written-over groups can be uniquely read at a later time to reproduce the incoming data information.

In another aspect of the invention, the apparatus relates to writing a plurality of times at the same group of bit positions (WITS) of a non-erasable digital memory. The apparatus features reading circuitry for reading the data stored at the bit positions and input circuitry for receiving new input data to be retrievably stored at the bit positions. A mapping circuit is responsive to the reading circuit and the input receiving circuit for generating a storeable codeword according to a mapping method, the newly generated codeword being greater than or equal to (as is defined hereafter) the word previously stored at the bit positions. The storeable codeword is then written, by a writing circuit, at the bit positions.

In particular embodiments of the invention, the mapping method can operate either according to a linear womcode or a tabular womcode as those coding methods are described hereinafter. In one particular embodiment of the invention, the mapping circuit employs a read only memory (ROM).

The method of the invention relates to writing a plurality of times at the same group of bit positions of a non-erasable digital memory. The method features the steps of reading the data stored at the bit positions and receiving input data to be retrievably stored at the bit positions. The method further features generating a storeable codeword in response to the read and received data, the storeable codeword being greater than or equal to the word previously stored at the memory bit positions. The method then features writing the storeable codeword at the bit positions, in effect overwriting the previously stored data.

Brief Description of the Drawings

Other objects, features, and advantages of the invention will become apparent from the following description of preferred particular embodiments of the invention taken together with the drawings in which:

Figure 1 represents a simple diagrammatic representation of a method for rewriting in a non-erasable memory;

Figure 2 is a table describing the mapping method used in Figure 1;

Figure 3 is a schematic block diagram of apparatus for rewriting in a non-erasable memory;

Figures 4A-4B are more detailed representations of a mapping element of Figure 2; and

Figure 5 is a table showing the optimum word length requirements for various groupings of word length in rewriting a non-erasable memory.

Description of Particular Preferred Embodiments

Non-erasable memories, such as digital optical disks, are typically viewed as a write-once memory wherein the capacity of the disk is equal to the number of positions on the disk at which a data bit can be stored. Such thinking however does not take into account the ability to reuse a disk, in the manner set forth below, which, in effect, increases the capacity of the disk substantially. The magnitude of the improvement varies depending upon the particular mapping method employed for reusing the disk and whether or not the memory being "reused" was originally written with that intent. As described below, very simple methods and apparatus can be employed to increase the capacity of a non-erasable medium by a factor of one-third. More complex methods and apparatus can be employed to obtain even more dramatic results.

A non-erasable memory or "write-once memory" is designated a "WOM". Each WOM consists of a number of "write-once bit positions" (called "WITS")j and each WIT initially contains a "zero" that can later be selectively, but irreversibly, overwritten with a "one". Typically, a large WOM can be divided into data blocks or sections that are used as needed, an index being provided for example on an associated magnetic disk, to keep track of the valid blocks. The magnetic disk can be eliminated by using a linked-list or tree-like data structure on the WOM itself to link obsolete blocks to their replacements; however, the effect of storing information on the WOM itself is to increase the cost in terms of access time.

By using mapping or coding methods according to the invention, a WOM can be "rewritten" many times

and correspondingly it will have a "bit capacity" which is much greater than the number of its bit positions or WITS. Many of the coding techniques which are disclosed later are both simple to implement and can have a significant impact on the effective cost of using the WOM.

Before describing the invention in more detail, a simple, but very effective, example can be employed to provide the basic understanding of the method for increasing the medium capacity, that is, for "reusing" or "rewriting" a non-ersable memory. Referring to Figures 1 and 2, as an example of a coding method to which the present invention is directed, consider a method which employs groups of three WITS or bit positions on the WOM to write two bits of data information twice. That is, by writing once, and then again, in three bit positions of a non-erasable media, one can represent first, the data of a first group of two bits (written once) and, at a later time, the data of a second group of two bits (overwritten upon the same three bit positions used to represent the first group of two bits).

Referring in particular to the table of Figure 2, the initial two-bit input data value, "x", is written the first time as a three bit position space (three WITS) "y" according to a function $y = r(x)$. Thereafter, data x_1 previously written as $y_1 = r(x_1)$, can be "rewritten" in accordance with new input data x_2 according to the function $y_2 = r'(x_2)$ for $x_2 = x_1$; and $y_2 = y_1$ for $x_2 = x_1$ (that is, if the new data x_2 is equal to the old data x_1, the pattern y_1 is not changed and $y_2 = y_1$). Note that $r'(x_2)$ will have ones wherever

$r(x_1)$ does (for any $x_2=x_1$) so that in going from r to r' one need only change zeros to ones.

The Boolean three-cube of Figure 1, wherein the lower level of the cube represents the first generation of "writing" and the upper level of the cube represents the second generation of "rewriting", graphically describes the writing sequence.

Decoding of the data is quite easy. The memory word always represents the two-bit value (b+c), (a+c), independent of whether the WOM has been written once or twice. The EXCLUSIVE OR Boolean operation represented by the sign "+" is a standard Boolean operation, is easily implemented, and is well-known to those skilled in the art.

According to the method described in connection with Figures 1 and 2, four data bits can be written in three WITS of the WOM. Thus the method described above increases the WOM's capacity by a factor of one-third.

Having introduced the concept of the method and apparatus, it is helpful to introduce some notation which makes the following more general discussion easier. The "weight" of a binary code word is defined to be the number of ones it contains. "x+y" will be used to denote the bit-wise EXCLUSIVE OR of the bit vectors "x" and "y" (which are both assumed to have the same length). A binary word $x = x_1...x_r$ is said to be "above" another binary word $y = y_1...y_s$ (denoted "x greater than or equal y"), if r = s and x_i is greater than or equal to y_i for all i between 1 and r. All

logs, that is log(x), denote the logarithm to the base 2 of x and the integer value of the logarithm is written log(x) .

A mapping method which uses n WITS to represent one of v values so that it can be written a total number of t times (that is, written once and changed t-1 times) is denoted as "$(v)^t$/n-womcode". The "/n" can be dropped for an optimal womcode, i.e. one with minimal n. The general case, where the number of values can change from generation to generation, is denoted by $[v_1,\ldots,v_t]$/n-womcode. The notation $w([v_1\ldots v_t])$ denotes the least n for which a $[v_1,\ldots,v_t]$/n-womcode exists. Similary, $w([v]^t)$ denotes the number of WITS needed for an optimal $[v]^t$ womcode.

It may seem paradoxical at first that $w([v]^t)$, less than $\log(v)\cdot t$ could exist. Intuitively the reason is that only the last value written needs to be accessible, that is, previous values may become irretrievable. In fact, if all previous values were always accessible, then the number of WITS required would be $\log(v)\cdot t$.

Referring to Figure 3, an apparatus for implementing the various mapping methods to be described herein has an input data group formation circuit element 10 for receiving, either serially or in parallel, input data on lines 12. The data group formation element can be, for example, a shift register designed to convert serial input data into input data groups of length k. The thus formed input data groups are provided over lines 14 to a mapping or coding circuit element 16. Element 16 can be, for example, a read only memory (ROM)

(Fig 4A) whose output provides both decoded data over lines 18 and coded data groups over lines 20. The input to the mapping element 16 must include not only the data groups over lines 14 but code words previously written (if any) which are provided from a media read circuit element 22 over lines 24. The media read element 22 reads data groups from a non-erasable medium 28. The media read element 22 operates in a clocked synchronism with a media write circuit element 26. The media write element 26 takes the coded groups of data available on lines 20 and writes them on the non-erasable medium 28, for example a digital optical disk. The synchronization between read element 22 and write element 26, through the coding or mapping element 16, enables the write element to overwrite (or rewrite) at the precise position of previously read WITS (corresponding to the input to the element 16 over lines 24) so that previously written data is overwritten in a known and predetermined manner.

As noted above, the coding element 16 can be for example a read only memory (ROM) 30 (Fig. 4A). The read only memory takes as its inputs, the input data groups on lines 14 and the read data groups on lines 24. These data inputs effectively address a word of the ROM and provide at the output of the ROM 30, a multibit word having two parts. One part, over lines 18, represents the decoded data corresponding solely to the media read input over lines 24. The other part, over lines 20, represents the newly coded data corresponding to both the read inputs over lines 24 and the new data inputs over lines 14. Referring to Figure 4B, a typical ROM map is shown for the mapping sequence and method illustrated in Figures 1 and 2. (Alternatively, other

hardware logic, well known in the art, for example a discrete component Boolean logic decoding circuit, can be employed to encode the input signals over lines 14 and 24 and to provide the required decoded and newly encoded output signals).

Referring to Figure 4B, it is interesting to note that when an already overwritten three-WIT data word is read, and because no further writing is permitted, the output over lines 20 can be any convenient value. This flexibility can be employed to reduce the size of the ROM by combining a smaller capacity ROM with various logic elements to provide the necessary mapping called for by the method represented in Figures 1 and 2.

While completely general observations regarding optimal coding methods have not beem completely worked out, the table of Figure 5 provides some available information regarding the minimal size for $w([2^k]^t)$. Obviously, $w([2^k]^1)$ equals k and $w([2^1]^t)$ equals t. The remaining numbers have been derived either by hand or using automated methods.

Attached to the application as Appendix 1 is a copy of a paper prepared by the inventors herein which describes in great detail various aspects of the reuse of a non-erasable memory. That paper details much of the theory which has been employed in determining the advantages of the invention and in determining several methods for generating mapping systems to enable efficient transformations to be derived. The substantive work described in that paper, with the exception of that described in Section VI, is the joint

work of the inventors herein although editorial comments regarding the paper have been received from other sources.

Examples of Other Methods for Forming Womcodes

The Tabular $[v]^t/n$-Womcode:

It is first assumed that t is greater than v. Further, let u denote an integer parameter to be chosen later (u to be chosen to be about equal to $\log(n)$). The $[v]^t/n$-womcode will have its $n = r \cdot s$ WITS considered as $r = (u+1)(v-1)$ rows of $s = \log(v)+t/(u\cdot(v-1))$ columns. Each row is divided into a $\log(v)$-WIT "header" field and an $(s-\log(v))$-WIT "count" field. The $\log(v)$-bit value represented by such a table is the sum (mod v) of the header fields of all rows that have an odd number of "1"s in their count fields.

Thus, to write a value x when the table currently represents y, it suffices to change a single "0" to a "1" in the count field of a row having the header $x-y \pmod v$. If every row with this header has all ones in its count field, we find a new row which is currently all zeros both in its header and count fields, change the header to the desired value, and change one bit of the count field. We can always find a new row up until $u(v-1)$ rows are completely "full", since there are only v-1 useful header values (the all zero value is useless as a header). Thus, we are guaranteed at least $u(v-1)\cdot(s-\log(v)) = t$ writes.

Since $n = r\cdot s = t+t/u+\log(v)(u+1)(v-1)$, choosing u equal to the integer value of $\log(t)$ implies $n = t+o(t)$. This code has a rate approximately $\log(v)$

which is less than log(n). The rate is the capacity per WIT of the womcode. With optimally chosen parameters, this code has a rate nearly equal to log(n) and is about twice as good as any other code described herein.

The "Linear" Womcode

This method employs the parameters v, t = 1+v/4, and n = v-1. The ith WIT is associated with the number i for i between 1 and v-1. The value represented by any pattern is the sum (mod v) of the numbers associated with the WITS in the one state. (As an alternate definition, useful when v equals a power of 2 , the pattern is interpreted as the EXCLUSIVE OR of the k-bit representations of the numbers associated with the WITS in the "1" state. For example, the $[2^2]^{2}/3$ womcode of example 1 is a Linear womcode, and the coded pattern abc can be decoded as $01\cdot a+10\cdot b+11\cdot c$.)

According to this code, so long as there are at least v/2 zeros, the W can be changed to represent a new value by changing at most two WITS. Thus, the mapping described above has a rate roughly equal to log(v)/4 which is approximately equal to log(n)/4.

A $[2^2]^{5}/7$ Womcode

A $[2^2]^{5}/7$ (a rate equal to 1.42 approximately) womcode has been developed by an ad hoc method. The pattern is denoted as a seven WIT word abcdefg. If the pattern has a weight equal to four or less, the value represented is $01\cdot c_{01}+10\cdot c_{10}+11\cdot c_{11}$, where c_{01} = 1 if and only if ab equals 10 or (ab = 11 and one of cd or ef is 01), c_{10} = 1 if and only if cd = 10 or (cd = 11 and one of ab or ef is 01), and c_{11} = 1 if and only if ef =

10 or (ef = 11 and one of ab or cd is 01). For example, the pattern 1101100 represents 10. At most, one of ab, cd, or ef will be 11 if another is 01. Otherwise, the pattern is interpreted as the EXCLUSIVE OR of ab, cd, ef, and gg. The first three writes change at most one WIT while the last two writes may change two WITS.

Additions, subtractions, deletions, and other modifications of the disclosed particular embodiments of the invention, including new coding or mapping methods, will be obvious to those practiced in the art and are within the scope of the following claims.

HOW TO REUSE A "WRITE-ONCE" MEMORY
(Preliminary Version)

Ronald L. Rivest
MIT Laboratory for Computer Science, Cambridge, Mass.
Adi Shamir
Weizmann Institute of Science, Rehovot, Israel

Abstract

Storage media such as digital optical disks, PROMS, or paper tape consist of a number of "write-once" bit positions (WITS); each wit initially contains a "0" that may later be irreversibly overwritten with a "1". We demonstrate that such "write-once memories" (WOMS) can be "rewritten" to a surprising degree. For example, only 3 wits suffice to represent any 2-bit value in a way that can later be updated to represent any other 2-bit value. For large k, $1.29...\cdot k$ wits suffice to represent a k-bit value in a way that can be similarly updated. Most surprising, allowing t writes of a k-bit value requires only $t+o(t)$ wits, for any fixed k. For fixed t, approximately $k\cdot t/\log(t)$ wits are required as $k \to \infty$. An n-wit WOM is shown to have a "capacity" (i.e. $k\cdot t$ when writing a k-bit value t times) of up to $n\cdot \log(n)$ bits.

I. Introduction

Digital optical disks (a variation of the "video disks" used to store analog video data) are an exciting new storage medium. A single 12-inch disk costing $100 can be used to store over 10^{11} bits of data — the equivalent of 40 reels of magnetic tape — and to provide access to any of it in 1/10 second. Such an order-of-magnitude improvement in the cost/performance of memory technology can have dramatic effects. (See [Bu80],[Mc81],[Go82].)

However, such capability is achieved at the cost of making the writing process irreversible. The disks are used as follows. Each disk is manufactured with a thin reflective coating of tellurium. To write on the disk, a laser is used to melt submicron pits in the tellurium at specified positions, changing those positions from their virgin "0" state to a "1" state. To read the disk, the laser (at low power) illuminates each position on the disk; the lower reflectivity of the pits is easily sensed.

The tremendous capacities and cheap cost per bit of digital optical disks provides a strong motivation to examine closely their one drawback — their "write-once" nature. The purpose of this paper is thus to explore the true capabilities of such "write-once memories" (or woms). Other familiar examples of woms are punched paper tape, punched cards, and PROMS (programmable read-only memories - in which the wits are microscopic fuses that can be selectively blown).

Large woms might naturally be used to store data that is more or less static: programs, documents,

pictures, data bases, or archival storage dumps. If the data requires updating, the wom can be replaced by a freshly written wom. A large wom can be divided into blocks that are used up as needed; a index on an associated magnetic disk can keep track of the valid blocks. The magnetic disk can be eliminated by using linked-list or tree-like data structures on the wom itself to link obsolete blocks to their replacements, at some cost in terms of access time.

More formally, we model a wom as an array of "write-once bits" (or wits) which are manufactured in a "0" state but which can later be independently but irreversibly transformed into a "1" state. (We understand that some of the current recorder/player designs for digital optical disks are not capable of selectively changing individual zero bits within a previously written block. However, this seems to be more a matter of engineering than of fundamentals.)

The main result of this paper is that by using appropriate coding techniques, a wom can be "rewritten" many times, and that its "bit-capacity" is much greater than the number of its wits. Many of the coding techniques proposed here are simple to implement, and can have a significant impact on the cost of using woms.

As an example of the kind of behavior we are interested in, the following coding scheme was a prime "movating example" for this research.

Lemma 1. Only 3 wits are needed to "write 2 bits twice".

Proof: We show how to represent a 2-bit value x in 3

wits so that it can later be changed to represent any other 2-bit value y. First, represent x with the pattern r(x) given in Table 1. Later, a value y($y \neq x$) can be written by changing the pattern r'(y). (If x = y no change is made). Observe that r'(y) will have ones wherever r(x) does, so that we need only change zeros to ones.

x	r(x)	r'(x)
00	000	111
01	100	011
10	010	101
11	001	110

Table 1. A $(2^2)^2/3$-Womcode

Decoding is easy: the memory word abc represents the 2-bit value $(b \oplus c),(a \oplus c)$, no matter whether the wom has been written once or twice.

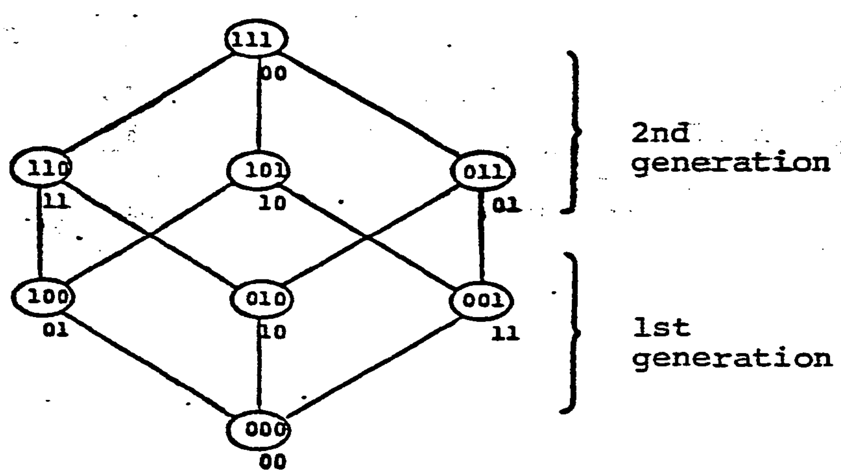


Figure 1. The $(2^2)^2/3$-womcode on the Boolean 3-cube

II. Notation

Let weight of a binary codeword be the number of ones it contains. Let $x \oplus y$ denote the bitwise XOR of the bit vectors x and y (assumed to have the same length). We say that an binary word $x = x_1 \ldots x_r$ is "above" another binary word $y = y_1 \ldots y_s$ (denoted $x \geq y$) if $r = s$ and $x_i \geq y_i$ for $1 \leq i \leq r$. Let log(x) denote the logarithm (base 2) of x (or, if the context requires an integer value, $\lceil \log_2(x) \rceil$). We use Z_v to denote the set $\{0,1,\ldots v-1\}$, and Z_2^n to denote the set of all binary words of length n. We say "$f(n) \approx g(n)$" if $\lim_{n \to \infty} f(n)/g(n) = 1$. (The variable being taken to the limit should be clear from the context.) We also let H(p) denote the "entropy function" $H(p) = p\log(1/p) + (1-p)\log(1/(1-p))$.

A coding scheme that uses n wits to represent one of v values so that it can be written a total of t times (i.e. written once and changed t − 1 times) we call a "$[v]^t/n$-womcode" (read: a "v t-times into n-wit womcode"). The "/n" may be dropped for an optimal wom-code (with minimal n) (read: an "optimal v t-times womcode"). The general case – where the number of values may differ from generation to generation – we call a "$[v_1,\ldots,v_t]/n$-womcode" (read: a "v_1 to v_t into n-wit womcode"); here the value stored on the i-th write may be any one of $[0,\ldots,v_i-1]$.

Let $w([v_1,\ldots,v_t])$ denote the least n for which a $[v_1,\ldots,v_t]/n$-womcode exists. Similarly, we use $w([v]^t)$ to denote the number of wits needed by an optimal $[v]^t$ womcode. We are interested in characterizing the behavior of $w([v]^t)$ for large values of v or t as well as finding "practical" coding schemes for small values of v or t.

It seems at first paradoxical that $w([v]^t) < \log(v)\cdot t$ could happen. Intuitively, the reason is that only the last value written needs to be accessible - previous values may become irretrievable. In fact, if all previously written values were always accessible then $\log(v)\cdot t$ wits would be required.

To make our definition of a womcode precise, we note that a $(v_1,\ldots,v_t)/n$ womcode can be defined to consist of the following parts (here let $v = \max(v_1,\ldots,v_t)$):

(1) An interpretation function α that maps each $x \in Z_2^n$ to a value $\alpha(x)$ in Z_v,

(2) An update function μ which gives for any $x \in Z_2^n$ and for any value y in Z_v, either "$\perp$" (i.e. undefined), or else a codeword $\mu(x,y) = z \in Z_2$ such that $\alpha(z) = y$ and $z \geq x$.

We say that α and μ define a correct $[v_1,\ldots,v_t]/n$-womcode if they "guarantee at least t writes" as defined below. In order to define this condition we first make the following auxiliary definitions.

An acceptable sequence $[i_1,\ldots,i_m]$ for a $[v_1\ldots,v_t]/n$-womcode satisfies the conditions that $0 \leq m \leq t$ and each i_j, $1 \leq j \leq m$, is in Z_{v_j}. Note that in particular the null sequence λ is acceptable.

We define the "write function" ρ mapping acceptable sequences to codewords (or $\perp$) as the

iteration of μ starting from the all-zero word 0^n (corresponding to the "initial state" of the wom) by the following equations:

$$\rho([i_1,\ldots,i_j]) = \begin{cases} 0^n & \text{if } j = 0 \\ \mu(\rho([i_1,\ldots i_{j-1}]),i_j), & \text{if } j \geq 1. \end{cases}$$

(We assume that $\mu(\perp,y) = \perp$ for all y.) We say that an acceptable sequence "arrives at x" if that sequence is mapped by ρ to x.

We say that a codeword x is "used in generation m," or "is an mth generation codeword" if there is an acceptable sequence of length m that arrives at x. A codeword x is said to be "unused" if no acceptable sequence of positive length arrives at x, otherwise we say x is "used".

If every codeword belongs to at most one generation we call the womcode "synchronized" - since all acceptable sequences that arrive at a codeword word arrive there "at the same time" (i.e. at the same generation). Otherwise the womcode is called "unsynchronized." With a synchronized womcode one can always determine how many generations have been written. Note that our $[2^2]^2/3$ womcode is not synchronized since 000 belongs to the zeroth, first, and second generations. We say that a womcode is "almost synchronized" if the all-zero word is the only codeword that belongs to more than one generation, and it belongs only to the zeroth and first generations.

The laminar womcodes are an interesting special case of the synchronized womcodes: a womcode is

laminar if it is synchronized and the weight of every (used) codeword determines its generation. (That is, no two codewords of different generations have the same weight.)

We say that the womcode defined by α and μ "guarantees at least t writes" if no acceptable sequence of length t arrives at $\perp$. This completes our formal definition of a $[v_1,...,v_t]/n$-womcode defined by α and μ; such a womcode is correct if it guarantees at least t writes.

We will often identify an interpretation $\alpha(x)$ with its binary representation. (For example, in a $[2^k]^t/n$-womcode each n-bit codeword represents a k-bit word.)

We would like to note that we initially studied only the $[2^k]^t/n$-womcodes, but that we have since seen enough interesting examples of womcodes of the more general form to warrant including the more general definition here.

We now introduce our three "complexity measures": P, I, and C.

Let P(t) denote the "penalty expansion factor" needed to guarantee t writes of values from Z_v, for large v, compared to that needed for just a single write:

$$P(t) = \lim_{v \to \infty} \frac{w([v]^t)}{\log(v)}. \qquad (1)$$

We will prove that $P(2) = 1.29...$ and $P(t) \approx t/\log(t)$.

Let $I(v)$ denote the asymptotic "incremental cost" of increasing t by one for a $[v]^t$-womcode:

$$I(v) = \lim_{t \to \infty} \frac{w([v]^t)}{t} \qquad (2)$$

We shall prove the surprising result that $I(v) = 1$.

We define the apparent capacity (in bits) of a $[v_1,...,v_t]/n$-womcode to be $\log(v_1...v_t)$; we denote this as $C([v_1,...v_t]/n)$. Similarly, let $C(n)$ denote the apparent capacity (in bits) of an n-wit memory field:

$$C(n) = \max\{\log(v_1...v_t) \mid w([v_1,...,v_t]) \leq n\}. \qquad (3)$$

We shall demonstrate that $C(n) = n\cdot\log(n)+o(n\cdot\log(n))$. As an auxiliary definition, we let $R([v_1,...v_t]/n) = C([v_1,...v_t]/n)/n$ denote the rate of the womcode. (This is just the capacity per wit of the womcode.)

III. Elementary Observations

Lemma 2.

$$w([v_1\cdot v_2]^t) \leq w([v_1]^t)+w([v_2]^t) \qquad (4)$$

Proof. Concatenate a $[v_1]^t$-womcode and a $[v_2]^t$-womcode to make a $[v_1\cdot v_2]^t/(w([v_1]^t))+w([v_2]^t))$ womcode. (Represent each value y in $Z_{v1\cdot v2}$ as an ordered pair (y_1,y_2), with $y_1 \varepsilon Z_{v1}$ and $y_2 \varepsilon Z_{v2}$. Use the womcodes to record y_1 and y_2 separately).

Lemma 3.

($w([v]^t)$ is subadditive in t.)

$$w([v]^{t_1+t_2}) \leq w([v]^{t_1}) + w([v]^{t_2}) \qquad (5)$$

Proof. Use side-by-side optimal $[v]^{t_1}$-and $[v]^{t_2}$-womcodes to represent the sum (mod v) of the values represented by the two subcodes. To update, change one subcode to represent the difference (mod v) of the new value to be represented and the value of the other subcode. This guarantees at least t_1+t_2 writes. (The alternative approach of writing the new value into one of the two subcodes would need extra wits to indicate which subcode was written last, unless the zero word is unused in one of the subcodes.)

The above lemmas (and $w([2^1]^1) = 1$) imply that $w([2^k]^t) \leq k \cdot t$.

For small values of k and t, we can derive $w([2^k]^t)$ as given in Table 2. (Obviously, $w([2^k]^1) = k$ and $w([2^1]^t) = t$.)

k \ t	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	3	5	6	7	
3	3	5	7			
4	4	6				
5	5	8				
6	6	9				
7	7					

Table 2. $w([2^k]^t)$

We do not know the exact values corresponding to the empty positions of the table.

The $[2^3]^{3}/7$ (rate 1.28...) and $[2^2]^{2}/3$ (rate 1.33...) womcodes indicated by the table are special cases of the general "linear" scheme presented in section V.

The $[2^2]^{5}/7$ (rate 1.42...) womcode indicated by the table is an ad hoc scheme; we show here how to decode a 7-wit pattern abcdefg. If the pattern has weight four or less, the value represented is $01 \cdot c_{01} \oplus 10 \cdot c_{10} \oplus 11 \cdot c_{11}$, where $c_{01} = 1$ iff ab = 10 or (ab = 11 and one of cd or ef is 01), $c_{10} = 1$ iff cd = 10 or (cd = 11 and one of ab or ef is 01), and $c_{11} = 1$ iff ef = 10 or (ef = 11 and one of ab or cd is 01). (For example, the pattern 1101100 represents 10. At most one of ab, cd, ef will be 11 if another is 01.) Otherwise the interpretation is ab $\oplus$ cd $\oplus$ ef $\oplus$ gg. The first three writes change at most one wit, while the last two might each change two wits.

The following notation for the size of the tail of a binomial distribution and a related inverse quantity will be useful:

$$\left(\!\!\binom{m}{h}\!\!\right) = \sum_{i=0}^{h} \binom{m}{i}. \tag{6}$$

$$\delta(v,m) = \min \left\{ h \;\middle|\; \left(\!\!\binom{m+h}{h}\!\!\right) \geq v \right\}. \tag{7}$$

Note that a $[v]^t/n$-womcode must have $n \geq m + \delta(v,m)$ if every first generation codeword must have at least m zeros. We derive a lower bound $Z(v,t)$ to $w([v]^t)$ by generalizing this observation:

$$Z(v,0) = 0 \text{ and} \tag{8}$$

$$Z(v,t+1) = Z(v,t) + \delta(v,Z(v,t)) \text{ for } t \geq 0. \tag{9}$$

Lemma 4.

$$w([v]^t) \geq Z(v,t) \tag{10}$$

Proof. By induction on t. The case $t = 0$ is trivial. A $[v]^{t+1}/n$-womcode must have at least $Z(v,t)$ zeros in every first generation codeword, and must turn on at least $\delta(v,Z(v,t))$ wits in the worst case on the first write to have v codewords in the first generation.

Corollary.

$$w([2^k]^t) \geq k + t - 1 \tag{11}$$

Note that $Z(2^k,1) = k$ and $Z(2^k, t+1) \geq Z(2^k,t) + 1$ for $k > 0$. The following lemma improves this result (by one).

Lemma 5.

$$w([2^k]^t) \geq k + t \text{ for } k \geq 2 \text{ and } t \geq 3. \tag{12}$$

Proof. Suppose to the contrary that a $[2^k]^t/(k+t-1)$-womcode existed for $k \geq 2$ and $t \geq 3$. Since $Z(2^k,1) = k$, the generation $t - 1$ codewords must each have weight less than t. On the other hand, if $t \geq 3$ then for every value $y \in Z_{2^k}$ there is a $t - 1$-st generation codeword x

with $\alpha(x) = y$ and weight at least $t - 1$. (For $t = 2$ the claim fails if the zero-weight word is in the first generation.) There must be at least $2^k - 1 \geq 3$ different values y associated with first-generation codewords of weight 1 or more. Thus for every value $y \in Z_{2^k}$ there is a $t - 1$-st generation codeword x with $\alpha(x) = y$ and weight exactly $t - 1$. But then no possible interpretation for the codeword 1^{k+t-1} is distinct from each of these values (required since the last k levels are "tight"). This contradiction proves the lemma.

IV. How many wits are needed for a fixed number of generations?

IV.A. How many wits are needed for two generations?

$$w([v]^2) = 1.293815...\log(v) \tag{13}$$

Proof. For any v, choose h to be $\delta(v, \log(v))$ and then chose n to satisfy:

$$n - h = \lceil \log(v) + \log\log(v) + 1 - \log\log(e) \rceil. \tag{14}$$

We will prove that $w([v]^2) \leq n$. Choose the first generation representations arbitrarily as distinct codewords with weight at most h, and ramdomly assign to the remaining $2^n - v$ codewords interpretations from Z_v. There are

$$(v)^{2^n - v} \tag{15}$$

ways to do this. How many ways do not guarantee two writes? Such a bad assignment must contain a first

generation codeword x and a value $y \in Z_v - \{\alpha(x)\}$ such that no codeword $z \geq x$ represents y. If we select x in one of v ways, select y in less than v - 1 ways, assign all codewords $z \geq x$ values different than $\alpha(x)$ and assign all other codewords arbitrary values, we will have overcounted the bad codes but examined no more than

$$v^2 \cdot (v-1)^{2^{n-h}} \cdot (v)^{2^n - v - 2^{n-h}}. \tag{16}$$

codes. Whenever (16) is less than (15) some "good" codes must exist. This happens when

$$v^2 \leq \left(\frac{v}{v-1}\right)^{2^{n-h}} \tag{17}$$

which will happen if

$$2\log(v) \leq 2^{n-h-\log(v)} \cdot \log(e) \tag{18}$$

which is implied by

$$n = h + \lceil \log(v) + \log\log(v) + 1 - \log\log(e) \rceil. \tag{19}$$

Thus (19) implies the existence of a $[v]^t/n$-womcode. Since $n \geq h + \log(v)$ (from Lemma 4), we conclude that for an optimal $[v]^2/n$-womcode

$$n = h + \log(v) + o(\log(v)). \tag{20}$$

Now the logarithm of the number of words of length n with at most h ones is

$$n \cdot H(h/n) + o(n), \text{ for } h \leq n/2. \tag{21}$$

(See [PW72, Appendix A], or [MS77, Ch. 10, §11].) Since there are t values in the first generation,

$$n \cdot H(h/n) + o(n) = \log(v) \qquad (22)$$

or (since $\log(v) = n - h + o(\log(v))$ and $n \leq 2 \cdot \log(v)$)

$$H(h/n) = \left(\frac{(n - h)}{n}\right) + o(1) \qquad (23)$$

The equation $H(p) = 1 - p$ has a solution at $p = 0.22709219\ldots$, so for an optimal womcode $h/n \approx .227\ldots$, or $\log(v) \approx n \cdot (1 - .227\ldots)$
or

$$n \approx 1.29381537\ldots\log(v) \qquad (24)$$

which was to be proved.

The random womcodes of the theorem will have an asymptotic rate of $2/1.29\ldots = 1.5458\ldots$, much better than the rate 1.33... womcode of lemma 1. However, we could not construct by hand a $[2^k]^2$-womcode of rate higher than 1.33.... Lemma 4 implies that such a scheme must have $k = 7$, $n = 10$ or $k \geq 9$. Using a computer we found a slightly more efficient method with rate 1.34....

The new scheme is a $[26]^2/7$-womcode (rate = 1.3429...). So a seven-track paper tape is "reusable" for writing just letters! Row i, column j of Table 3 gives the value (a letter) of the 7-bit string with binary value $i * 32 + j$. The first-generation is in

upper case. Thus a "T" (0011000) is made into an "h" by changing bits 1, 2, and 5 (to obtain 1111100). We were unable to find a $[27]^2/7$-womcode or to prove one doesn't exist, although we can prove that a $[29]^2/7$ womcode doesn't exist.

	00000000001111111111222222222233 01234567890123456789012345678901
0	AHGGFYLwEZYrXfpnDWVzUdjoTwkeltdu
1	CSRcQiozPpihuexyOzsjsniwvcqgfkbm
2	BNMzLbgmKutbngfwJwrhkvxymjpsoqci
3	Ikmqlckuwteosdjvubdfgetpyxnlhrza

Table 3. A $([26]^2/7)$-womcode

IV.B. What is P(t)?

By reasoning similar to that of the proof of Theorem 1, we derived the following estimates for P(t). Note how closely the estimates are to t/log(t).

t	P(t) (est.)	t/log(t)
1	1.000	----
2	1.294	2.000
3	1.549	1.893
4	1.783	2.000
5	2.003	2.153
10	2.983	3.010
20	4.668	4.628
50	8.960	8.859
100	15.191	15.051
200	26.346	26.164

Table 4. P(t)(est.) vs. t/log(t)

To demonstrate our main result that $P(t) = t/\log(t)$, we define an upper bound to $w([v]^t)$ which is asymptotically equal to our lower bound $Z(v,t)$ of Lemma 4, to within a small additive term.

Theorem 2. For fixed t and v sufficiently large, a sufficient condition for the existence of a $[v]^t/n$-womcode is the existence of t numbers l_i, $1 \leq i \leq t$, such that

(a) $t \cdot \log(v) \geq n \geq l_1 \geq l_2 \geq \cdots \geq l_t \geq 0$,

(b) $\binom{n}{l_1} \geq v$,

(c) $\binom{l_i}{l_{i+1}} \geq v \cdot (t + 1) \cdot \log(v)$, for $1 \leq i \leq t$.

Proof.

We prove the existence of a $[v]^t/n$ womcode in which all i-th generation codewords contain exactly l_i zeros. Condition (b) implies that there are enough codewords with l_1 zeros for the v values of the first generation. We now show (by a counting argument) that for all i, $1 \leq i < t$ it is possible to assign interpretations to the codewards with l_{i+1} zeros for the i + 1-st generation in such a way that for every j, $0 \leq j < v$, every codeword with l_i zeros is below some codeword with l_{i+1} zeros that has been assigned interpretation j.

The total number of ways in which the $\binom{n}{l_{i+1}}$ codewords with l_{i+1} zeros can assigned values is:

$$v^{\binom{n}{l_{i+1}}}. \qquad (25)$$

We can overcount the number of "bad" ways of assigning interpretations to the codewords with l_{i+1} zeros (assuming we have already assigned interpretations to the earlier generations), in the following way. Choose a codeword x with l_i zeros, choose a "missing value" $y \varepsilon Z_v$, assign the $\binom{l_i}{l_{i+1}}$ codewords with l_{i+1} zeros above x with the v - 1 remaining values (other than y), and assign the other codewords with l_{i+1} zeros arbitrary interpretations. The number of "bad" ways is thus at most:

$$\binom{n}{l_i} \cdot v \cdot (v-1)^{\binom{l_i}{l_{i+1}}} \cdot v^{\binom{n}{l_{i+1}} - \binom{l_i}{l_{i+1}}} \qquad (26)$$

A "good" way must exist whenever (26) is less than (25). By simplifying this inequality, we get:

$$\binom{n}{l_i} \cdot v \cdot \left(1 - \frac{1}{v}\right)^{\binom{l_i}{l_{i+1}}} < 1. \qquad (27)$$

Since $n \leq t \cdot \log(v)$ (otherwise the existence of the desired womcode is trivial), $\binom{n}{l_i} \leq v^t$, and thus it is enough to prove:

$$v^{t+1} \cdot \left(1 - \frac{1}{v}\right)^{\binom{l_i}{l_{i+1}}} < 1. \qquad (28)$$

By condition (c), $\binom{l_i}{l_{i+1}} \geq v \cdot \log(v) \cdot (t + 1)$, and thus it suffices to prove that:

$$v^{t+1} \cdot \left(1 - \frac{1}{v}\right)^{v \cdot \log(v) \cdot (t+1)} < 1. \quad (29)$$

But for large enough v, $\left(1 - \frac{1}{v}\right)^{v}$ approaches $1/e$, and thus the left hand side is approximated by:

$$v^{t+1} \cdot e^{-\log(v)\cdot(t+1)}, \quad (30)$$

which approaches zero as v goes to infinity.

To find the smallest (or nearly smallest) n for which the existence of a $[v]^{t}$-womcode is guaranteed by the theorem, the numbers l_i should be chosen in reverse order (from l_t to l_1). The last two numbers can be chosen as:

$$l_t = \frac{\log(v)}{2} + c \log \log(v), \quad (31)$$

where c is any constant greater than 1, and

$$l_{t-1} = \log(v) + 2c \log \log(v), \quad (32)$$

since

$$\binom{l_{t-1}}{l_t} = \binom{2l_t}{l_t} > \frac{2^{2l_t}}{2l_t} = \frac{v\cdot(\log(v)^{2c})}{\log(v) + 2c \log \log(v)}. \quad (33)$$

Since $c > 1$ and t is fixed, this becomes larger than $v \cdot \log(v) \cdot (t + 1)$ for v sufficiently large. The other l_i's can chosen as the smallest numbers satisfying condition

(c) of the theorem. Finally, n can be chosen as the smallest number satisfying $\binom{n}{l_i} \geq v$.

We now proceed to analyze the performance of the womcodes described above, in order to show that their performance is asymptotically equal to that of the lower bound we proved in Lemma 4. Then we prove our main theorem (theorem 4) that $P(t) \simeq t/\log(t)$.

We first introduce some necessary notation. Let

$$\delta'(v,m) = \min\{h \mid \binom{m+h}{h} \geq v\}. \qquad (34)$$

(Note the similarity to the definition of δ in (7).) Then we define:

$$Y_u(v,0) = \frac{\log(v)}{2} + c \log \log(v), \qquad (35)$$

$$Y_u(v,1) = \log(v) + 2c \log \log(v), \text{ and} \qquad (36)$$

$$Y_u(v,t+1) = Y_u(v,t) + \delta'(v \cdot u \cdot \log(v), Y_u(v,t)), \quad \text{for } t \geq 1. \qquad (37)$$

For convenience in the next theorem, we define $Y(v,t)$ to be $Y_{\log(v)}(v,t)$; note that for large v, $Y(v,t) \geq Y_{t+1}(v,t)$. From this definition it follows that for v sufficiently large,

$$w([v]^t) \leq Y(v,t), \qquad (38)$$

since $l_i = Y_{t+1}(v, t - i)$ for $1 \le i \le t$ and $n < Y_{t=1}(v,t)$ in the construction of last theorem.

Theorem 3. For $t \geq 1$,

$$\lim_{v \to \infty} \frac{Y(v,t)}{Z(v,t)} = 1. \tag{39}$$

Proof.

By induction on t. The case t = 1 is trivial. By comparing the forms of the definitions of Y and Z, we see that it is enough to prove:

$$\lim_{v \to \infty} \frac{\delta'(v\cdot(\log(v))^2, m')}{\delta(v,m)} = 1, \tag{40}$$

where $m = Z(v,t - 1)$ and $m' = Y(v,t - 1)$. We observe that

$$\delta(v,m) \le \delta'(v,m) \le \delta(v,m) + 1 \tag{41}$$

if $m \geq \log(v)$ (since that implies that $\delta(v,m) \le (m/2)$). (Note that in (40) both m and m' are $\geq \log(v)$.) Thus we can replace δ' by δ in (40). Furthermore, $\delta(v,m)$ is a decreasing function of m, so that we can also replace m' by the smaller value m in (40). In a similar vein, it is simple to show that $m \geq \log(v)$ implies that

$$\delta(v\cdot(\log(v))^2, m) \le \delta(v,m) + 2 \log \log(v), \tag{42}$$

and a little more complicated to show that $\log(v) \le m \le \gamma \cdot \log(v)$ implies that

$$\delta(v,m) \geq \log(v)\cdot(2\gamma \cdot H^{-1}(1/2\gamma)) \qquad (43)$$

Combining these observations leads to the desired result.

Theorem 4. $P(t) \simeq t/\log(t)$.

Proof.

Let $n_t = Z(v,t)$. Then we must have:

$$v \leq \left(\!\!\binom{n_t}{n_t - n_{t-1}}\!\!\right) \simeq 2^{n_t \cdot H(n_{t-1}/n_t)} \qquad (44)$$

so we derive

$$H(n_{t-1}/n_t) \simeq \log(v)/n_t. \qquad (45)$$

We consider $H(p)$ near $p = 0$ using the fact that $H(p) = H(1 - p)$:

$$1 - n_{t-1}/n_t \simeq H^{-1}(\log(v)/n_t). \qquad (46)$$

Near $p = 0$, $H(p) = p\cdot\log(1/p)$, so $H^{-1}(y) = -y/\log(y)$:

$$1 - \frac{n_{t-1}}{n_t} \simeq \frac{-\log(v)/n_t}{\log(\log(v)/n_t)} \qquad (47)$$

$$n_t - n_{t-1} \simeq -\log(v)/\log(\log(v)/n_t) \qquad (48)$$

$$\frac{dn_t}{dt} \simeq \frac{-\log(v)}{\log(\log(v)/n_t)} \qquad (49)$$

$$dt \approx \frac{\log(n_t/\log(v))\cdot dn_t}{\log(v)} \qquad (50)$$

$$t \approx \left(\frac{n_t}{\log(v)}\right) \log \left(\frac{n_t}{\log(v)}\right) \qquad (51)$$

$$\frac{n_t}{\log(v)} \approx P(t) \approx \frac{t}{\log(t)} \qquad (52)$$

As a consequence of Theorem 4, for fixed t and large v, an optimal $[v]^t$ womcode will have a rate approximately equal to log(t), with the approximation improving as t increases.

V. What is I(v)?

In this section we demonstrate that I(v) = 1 for any v, using a "tabular" womcode. We also present a "linear" womcode that - while it only shows that I(v) $\leq$ 4, generalizes nicely our $[2^2]^2/3$-womcode.

V.A. The Tabular $[v]^t/n$-Womcode

We assume here that t > v. Let u denote an integer parameter to be chosen later (imagine that u is about log(n)). Our $[v]^t/n$-womcode will have its $n = r\cdot s$ wits considered as $r = (u + 1)(v - 1)$ rows of $s = \log(v) + t/(u\cdot(v - 1))$ columns. Each row is divided into a log(v)-wit "header" field and an (s - log(v))-wit "count" field. The log(v)-bit value represented by such a table is the sum (mod v)) of the header fields of all rows that have an odd number of "1"s in their count fields.

To write a value x when the table currently represents y, it suffices to change a single "0" to a "1" in a row with header $x - y \pmod{v}$. If every row with this header has all ones in its count field, we find a new row which currently is all zeros both in its header and count fields, change the header to the desired value, and change one bit of the count field. We can always find a new row up until $u(v - 1)$ rows are completely "full", since there are only $v - 1$ useful header values. (The all-zero value is useless.) Thus we are guaranteed at least $u(v - 1) \cdot (s - \log(v)) = t$ writes.

Since

$$n = t + t/u + \log(v)(u + 1)(v - 1),$$

by choosing $u = \lfloor \log(t) \rfloor$ implies that $n = t + o(t)$.

This code has rate approximately $\log(v) < \log(n)$. With optimally chosen parameters, this code has a rate nearly $\log(n)$, about twice as good as any other code presented in this paper.

V.B. The "Linear" Womcode

This scheme has parameters v, $t = 1 + v/4$, and $n = v - 1$. The i-th wit is associated with the number i, for $1 \leq i < v$. The value represented by any pattern is the sum (mod v) of the numbers associated with wits in the "1" state. (An alternative definition, useful when $v = 2^k$, interprets the pattern as the XOR of the k-bit representations of the numbers associated with wits in the "1" state. For example, in our $[2^2]^2/3$-womcode the pattern abc can be decoded as $01 \cdot a \oplus 10 \cdot b \oplus 11 \cdot c$.)

We now show that - as long as there are at least $v/2$ zeros - we can change the wom to represent a new value by changing at most two wits. Let z denote the difference (modulo v) of the new value desired, y, and the current value represented, x. If the bit associated with z is now "0" we can simply change it to a "1". Otherwise let S denote the set of numbers associated with wits which are currently zero, and let T denote the set $\{z - x \pmod{v} \mid x \in S\}$. Since $|S| = |T| \geq v/2$ and $|S \cup T| < v - 1$ (zero is in neither set), the set $S \cap T$ must be nonempty. Their overlap indicates a solution to the equation $z = x_1 + x_2 \pmod{v}$ where x_1 and x_2 are elements of S.

The code described above thus has rate roughly $\log(v)/4 \approx \log(n)/4$.

The linear womcode described above may have to stop when there are as many as $(v/2) - 1$ zeros left (which can happen after as few as $1 + (v/4)$ writes). The following trick allows one to keep going a while longer. Divide the n-wit field into $n/3$ blocks of size 3. By writing additional "1"s if necessary, force each block to have either one "1" or three "1"s exactly. Those "bad" blocks having no zeros remaining are now considered as deleted, while each of the remaining "good" blocks can be used to store one bit (by changing one of its wits - the other one is left untouched to indicate that the block is a good block and not a deleted block). With at least $(n-1)/2$ zeros remaining we are guaranteed of getting at least $n/12 - 1$ "good" blocks. The recurrence: $t(n) = n/4 + t(n/12)$ has the solution $t(n) = 3n/11 + O(1)$, indicating that this trick can increase the number of writes we can achieve by a

factor of 12/11 (from n/4 to 3n/11 writes). At the moment this coding trick is also the best general scheme we know of for making use of a "dirty" WOM that may have been written before in an arbitrary manner (i.e. without any thought of using some sort of "womcoding" for better utilization).

VI. What is C(n)?

The schemes presented in the last section can be used to show that $C(n) = n \cdot \log(n) + o(n \cdot \log(n))$.

Theorem 5.

$$C(n) \geq n \log(n) + o(n \log(n))$$

Proof. For a given large memory size n, we can use the "tabular" scheme of section V.A. and choose parameters: $\log(v) = \lfloor \log(n) - 2 \log\log(n) \rfloor$ with $r = \lceil \log\log(n) \rceil \cdot (v - 1)$ rows of length $s = \lfloor n/r \rfloor$. (We will "waste" $n - rs$ wits.) As before, the total number of writes possible is $n - o(n)$, proving the theorem.

It is also possible to show that this result is "best possible":

Theorem 6.

$$C(n) \leq n \cdot \log(n)$$

Proof. (Intuitively, changing one wit out of n should provide at most log(n) bits of information.) Consider any $[v_1, \ldots, v_t]/n$-womcode. The n-wit field can undergo at most $(t + 1)^n < n^n$ "histories" as it progresses from its first state (all "0"s) to its final state (perhaps all "1"s), since we can describe the history by

specifying for each of the n wits that it either always remains "0" or that it is turned to a "1" during one of the t write operations. On the other hand, the womcode has at least $v_1 \ldots v_t$ different acceptable sequences of length t to handle, each of which must have its own history. The theorem follows.

VI. Other Womcodes

Several of our colleagues have become intrigued by the problem of designing high-rate womcodes, and have graciously consented to our sketching or referencing their preliminary results here.

- Prof. David Klarner (Dept. Math, SUNY, Binghampton), has created an elegant $[5]^{3}/5$ (rate 1.39...) cyclic womcode, which works as follows. The first value is represented by 10000 in the first generation, either 01001 or 00110 in the second generation, and one of 01111, 10110, or 11001 in the third generation. The other four values are handled similarly, using cyclic rotations of the words given for the first value. (Since n is prime all the cyclic rotations are distinct.)

- David Leavitt (an undergraduate working with Prof. Spyros Magliveras, Dept. Math, Univ. Nebraska at Lincoln), has found an even more efficient $[7]^{4}/7$ (rate 1.60...) cyclic womcode by extending Klarner's technique. (To appear.)

- James B. Saxe (a graduate student at CMU) has created the following beautiful (n/2,n/2 -

1,...,1]/n womcode (rate asymptotically log(n)/2), where the two halves of each codeword are the same except that the left half has an extra "1" bit. The value represented is the number of zeros in the left half to the left of the extra bit. Each update (except the first), changes exactly two wits - one in each half.

- Saxe also suggested the following marvelous recursive womcode, which uses $n = 2^k$ wits, and changes exactly one wit per write. Using $\int(n)$ to denote the capacity of Saxe's code, we shall see that $\int(2^k) = k \cdot 2^{k-1}$, using the base case $\int(1) = 0$, giving a rate of log(n)/2. Partition the n wits into n/2 pairs. With the first n/2 writes we turn on at most one bit in each pair, and obtain capacity $\int(n/2) + n/2$ by using the code recursively on the n/2 pairs, and getting an extra bit/write using the parity of the number of pairs whose left bit is on. For the second n/2 writes we obtain capacity $\int(n/2)$ recursively on the pairs by turning on their second bits as needed. The recurrence $\int(n) = n/2 + 2\int(n/2)$ gives the desired result.

- Saxe has also created a [65,81,63]/12 (rate 1.52...) womcode that can be improved to a [65,81,64]/12 (rate 1.53...) womcode if the "generation number" is externally available when decoding.

VII. Discussions and Conclusions

The results presented in this paper provide

much information about the nature of the function $w([v]^t)$. On the basis of the evidence so far, we conjecture that

$$w([v]^t) \simeq \max(t, \frac{\log(v)\cdot t}{\log(t)})$$

for all large v and t. We expect that this result should follow in a more-or-less straightforward manner from the results and techniques given here, but we have not as yet worked through a detailed demonstration. (Exercise for the reader: prove that $w([2^k]^k) = \theta(k^2/\log(k))$.)

The relationship between womcodes and error-correcting codes are interesting: we can view a womcode as a situation where the channel is assymmetric (only $0 \longrightarrow 1$ errors occur), and where the transmitter knows where the errors will occur before he has to choose a codeword. Of course, there are still many differences, since the objective of womcoding is to allow many "messages" to be sent and the codeword for one message determines what the "errors" are for the next message.

A more closely related problem may be that of devising codes for random-access memories that have "stuck bits". Heegard [He81] has some recent work in this area. Again, however, the problem seems intrinsically different.

Some interesting work has been done on Turing machines that have "nonerasing" work tapes (e.g. see

[Mi67]), which is peripherally related to the research reported here.

We note that our formulation of the problem requires that the decoding scheme for an $[v]^t/n$-womcode provide a unique interpretation for each possible pattern of the n wits, independent of how many generations have been written. In some cases the current "generation number" might also be available as input to the decoding scheme at no extra cost (in wits). While this variation might permit some minor performance improvements in some instances, it remains an open question as to how much this additional information might help.

A number of questions need further investigation:

- What if there is some restriction on the kinds of updates that may occur? (For example, what if y can replace x only if y is numerically greater than x?)

- What advantages are there to representing a different number of values at each generation?

- What is the complexity of the decoding and updating algorithms for the best codes?

- How can these coding schemes be adapted to handle the possibility of errors occurring on the wom?

- If the underlying storage medium is viewed as storing a modulated digital signal rather than

a sequence of bits, what kind of "womcoding" should be used to allow updating yet to maximize bandwidth while minimizing modulation frequency? (See [Br81], [HG69]).

- What can be said about the average-case behavior of womcodes?

- What if the storage elements had more than two possible states, and had a complicated dag that described the set of legal state transitions?

- What truly practical womcodes exist?

Acknowledgements

We would like to thank Eric Brown, Abbas El Gamal, David Klarner, David Leavitt, Andrew Odlyzko, Michael Rubin, Jim Saxe, and Michael Sipser for their helpful comments and discussions.

References

[Br81] Brown, Eric Stewart. Digital Data Bases on Optical Videodisks. Bachelor of Science Thesis. (MIT, May 1981)

[Bu79] Bulthuls, K., M. Carasso, J. Heemskerk, P. Kivits, W. Kleuters, and Pl Zalm, "Ten Billion Bits on a Disk" IEEE SPECTRUM (Aug. 1979), 26-33.

[Bu80] "Videodiscs: A Three-Way Race for a Billion-Dollar Jackpot" Business Week (July 7, 1980), 72-81.

[Ga68] Gallagher, R.G. Information Theory and Reliable Communication. (Wiley 1968).

[Go82] Goldstein, C.M., "Optical Disk Technology and Information," Science 215, No. 4534 (12 February 1982), 862-868.

[He81] Heegard, C. "Capacity and Coding for Computer Memory with Defects," Ph.D. Thesis, Stanford University Dept. of Statistics Technical Report No. 45, (May 1981).

[HG69] Hecht, M. and A. Guida, "Delay Modulation", Proc. IEEE, (Letters), (July 1969).

[MS77] MacWilliams, F.J., and N.J.A. Sloane, The Theory of Error-Correcting Codes (North Holland, Amsterdam, 1977).

[Mc81] McLeod, J. "Optical Disks loom as replacement for Tape" Electronic Design (Sept. 30, 1981), 97-103.

[Mi67] Minsky, M. Computation: Finite and Infinite Machies. (Prentice-Hall, 1967)

[PW72] Peterson, W. and E. Weldon Jr. Error-Correcting Codes. (MIT Press, 1972).

What is claimed is:

1. Apparatus for reusing a non-erasable memory medium comprising

means for reading successive groups of digits from said non-erasable memory medium, and

means responsive to said read groups of data from said reading means and to incoming input data information for writing over each successive group according to a predetermined plan so that said written over groups can be uniquely read to generate said incoming input data information.

2. Apparatus for writing a plurality of times in a non-erasable memory comprising

means for receiving incoming digital data and for forming sequential groups from said received data,

means for reading successive groups of digital data from successive groups of positions in the non-erasable memory,

means responsive to each said formed sequential group and to a respective read successive group from memory for generating successive new coded groups of digital data, and

means for sequentially writing into said memory each said coded group of data at a said group of positions corresponding to said respective read group of data,

whereby said coded groups of data can be later read and uniquely decoded to generate said formed sequential groups.

3. The apparatus of claim 2 wherein said non-erasable memory, prior to storage of any data, is

comprised in its data storag area a succession of "zeros" and wherein data is stored by converting one or more of said "zeros" to "ones", and further wherein said responsive means comprises means for writing a non-decreasing number of "ones" as said groups of positions are written over.

4. Apparatus for writing a plurality of times at the same group of bit positions of a non-erasable digital memory comprising

means for reading the data stored at said bit positions,

means for receiving input data to be retrievably stored at said bit positions,

means responsive to said reading means and said receiving means for generating a storeable codeword, said codeword being greater than or equal to the word previously stored at said bit positions, and

means for writing said storeable codeword at said bit positions.

5. The apparatus of claim 4 further comprising

means connected to said reading means for uniquely converting said read data to output data.

6. The apparatus of claim 4 wherein said responsive means comprises

means for generating said storeable codeword according to a linear womcode.

7. The apparatus of claim 4 wherein said responsive means comprises

means for generating said storeable codeword according to a tabular womcode.

8. The apparatus of claim 4 wherein said r sponsive means comprises

a read only memory element responsive to said reading means and said receiving means for generating said storeable codeword.

9. The apparatus of claim 8 wherein said read only memory further converts said read data to a decoded data word.

10. A method for reusing a non-erasable memory medium comprising the steps of

reading successive groups of digits from said non-erasable memory medium, and

thereafter writing over each successive group in accordance with the data values of new incoming data information and the read groups from said memory according to a predetermined plan,

whereby said written over groups can be uniquely read to generate said incoming data information.

11. A method for writing a plurality of times in a non-erasable memory comprising the steps of

receiving input digital data;

grouping said received input data into sequential groups of input data;

reading from said memory successive groups of stored digital data,

forming successive groups of write digital data from said stored digital data and said input data sequential groups, said groups of write data being uniquely decodable to generate said input digital data, and

writing said groups of write data at memory positions corresponding to respective ones of said read groups of data in a one-to-one relationship.

12. The method of claim 11 further comprising the step of repeating said receiving, grouping, reading, forming and writing steps for writing a next sequence of input data in said non-erasable memory.

13. A method for writing a plurality of times at the same group of bit positions of a non-erasable digital memory comprising the steps of

reading the data stored at said bit positions,

receiving input data to be retrievably stored at said bit positions,

generating a storeable codeword in response to said reading and receiving steps, said codeword being greater than or equal to the word previously stored at said bit positions, and

writing said storeable codeword at said bit positions.

14. The method of claim 13 further comprising the step of

converting said data read during said reading step to output data.

15. The method of claim 13 wherein said generating step comprises the step of generating said storeable codeword according to a linear womcode.

16. The method of claim 13 wherein said generating step comprises the step of generating said storeable codeword according to a tabular womcode.

17. Th method of claim 13 wherein said generating step comprises generating said storeable womcode using a read only memory element.

18. The method of claim 17 wherein said generating step comprises the step of converting said read data to decoded output data words.

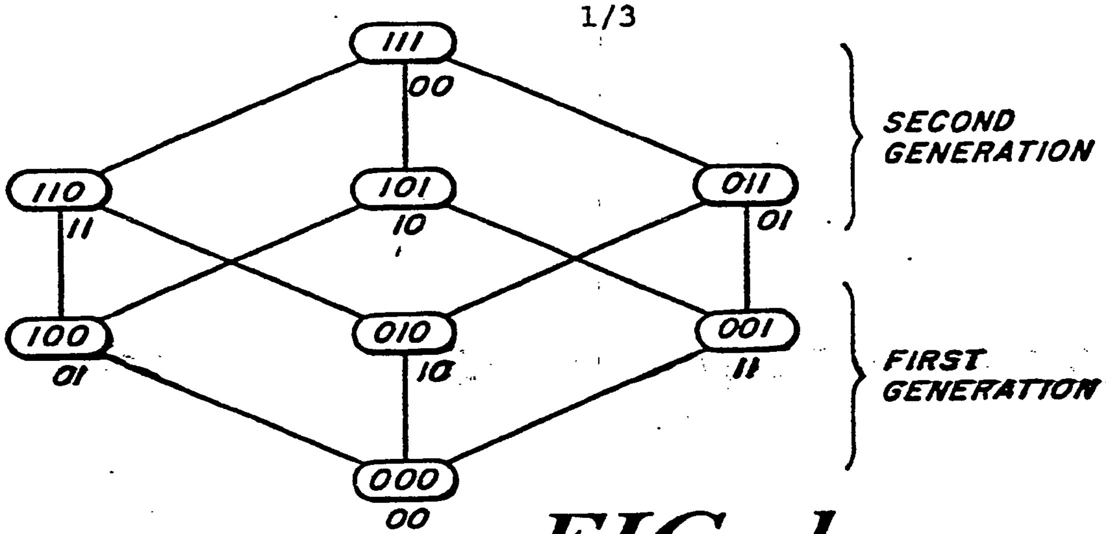


FIG. 1

x	r(x)	r'(x)
00	000	111
01	100	011
10	010	101
11	001	110

FIG. 2

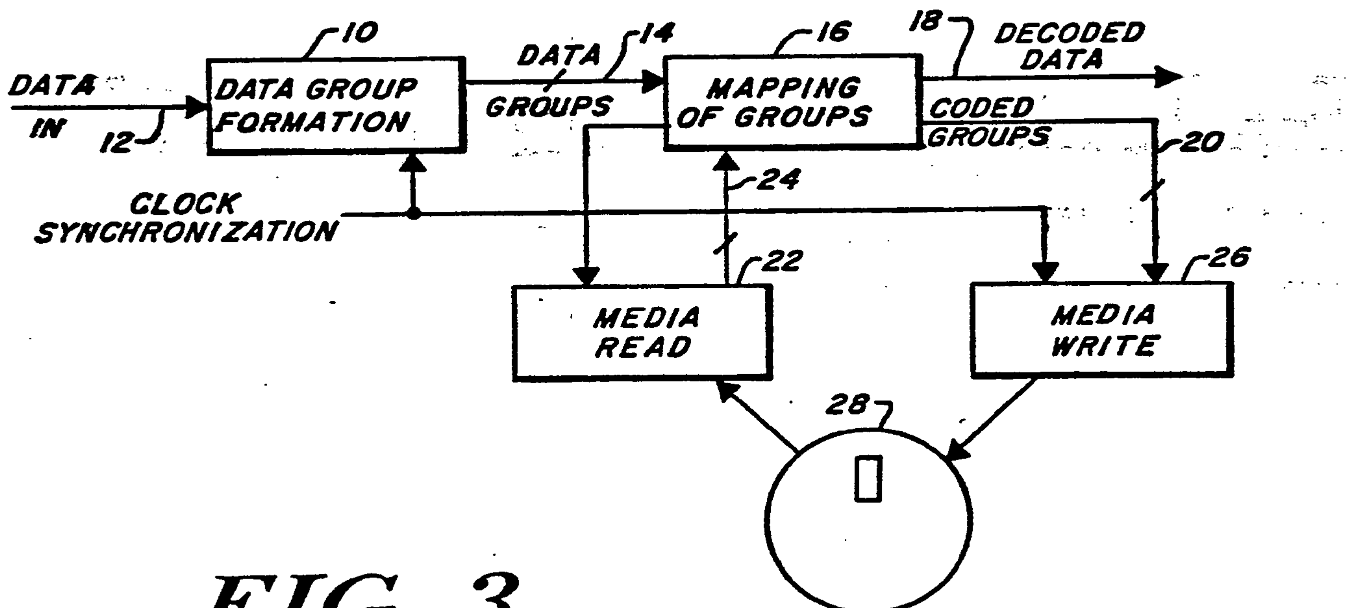


FIG. 3

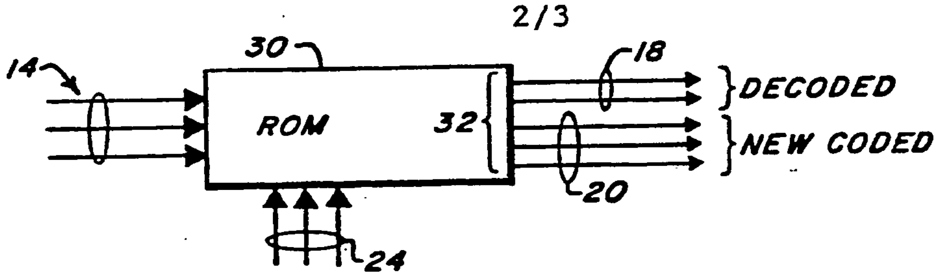


FIG. 4A

ROM LINES 14 (x)	INPUT LINES 24	ROM LINES 18	OUTPUT LINES 20 r(x) OR r'(x)
00	000	00	000
00	100	01	111
00	010	10	111
00	001	11	111
01	000	00	100
01	100	01	100
01	010	10	011
01	001	11	011
10	000	00	010
10	100	01	101
10	010	11	010
10	001	10	101
11	000	00	001
11	100	01	110
11	010	11	110
11	001	10	001
00	111	00	XXX
00	011	01	XXX
00	101	10	XXX
00	110	11	XXX
01	111	00	XXX
01	011	01	XXX
01	101	10	XXX
01	110	11	XXX
10	111	00	XXX
10	011	01	XXX
10	101	10	XXX
10	110	11	XXX
11	111	00	XXX
11	011	01	XXX
11	101	10	XXX
11	110	11	XXX

FIG. 4B

k \ t	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	3	5	6	7	
3	3	5	7			
4	4	6				
5	5	8				
6	6	9				
7	7					

$W(2^k, t)$

FIG. 5

INTERNATIONAL SEARCH REPORT

International Application No PCT/US82/00564

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [3]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int. Cl. G 06 F 7/26; G 11 C 17/00

U.S. Cl. 364/200, 900; 365/94, 189

II. FIELDS SEARCHED

Minimum Documentation Searched [4]

Classification System	Classification Symbols
U. S.	364/200, 900; 365/94, 120, 127, 189, 200, 215; 369/54, 58, 59

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [5]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [14]

Category *	Citation of Document, [16] with indication, where appropriate, of the relevant passages [17]	Relevant to Claim No. [18]
A	US, A, 3,609,708, Cragon et al, 28 September 1971	
A	US, A, 3,638,185, Dell et al, 25 January 1972	
A	US, A, 3,897,626, Beausoleil, 05 August 1975	
A	JP, A, 55-113137, Etsuno, 01 September 1980	

* Special categories of cited documents: [15]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search [2]	Date of Mailing of this International Search Report [2]
11 August 1982	20 AUG 1982
International Searching Authority [1]	**Signature of Authorized Officer [20]**
ISA/US	Stuart N. Hecker Stuart N. Hecker

Form PCT/ISA/210 (second sheet) (October 1981)